

Remarks

Claims 1-30 remain in the application. Please reconsider the application in view of the following remarks:

Patentability of Claims 1-30 over Sim, McCauley, or Chan

The Office Action rejects claims 1-30 as being unpatentable under 35 U.S.C. § 102 (e) over Sim, U.S. Patent 5,213,880, ("Sim"). The Office Action rejects claims 1, 20, and 22 as being unpatentable under 35 U.S.C. § 102 (e) over McCauley, U.S. Patent 6,263,392, ("McCauley"). The Office Action also rejects claims 1-3, 5-19 and 21-30 as unpatentable under 35 U.S.C. § 102 (e) over Chan, et al., U.S. Patent 5,991,546, ("Chan"). All rejections are respectfully traversed. For a 102(e) rejection to be proper, the cited art must show each and every element as set forth in a claim. (*See* MPEP § 2131.)

Patentability over Sim

Claim 1 is directed to a system for mapping an input device's controls to an application and recites in part, "an application program interface that receives calls from the application, the application program interface including a call that creates an association between actions in the application and the controls on the input device, wherein creating the association comprises considering semantics related to the actions in the application."

Sim fails to teach or suggest at least this feature of claim 1.

The Office Action asserts that the recited arrangement is disclosed in the following passages:

A pointer for the data structure is used to keep track of the multiple button keymapping of the game pad 100. The structure information is saved as bin format in the keymap files 362. Once the translator module 368 and the keymap files 362 are stored in memory, the game pad 100 is able to control any application module 366 using the DirectInput interface 430. In the preferred embodiment, after installation of the translator module 368, an icon is shown in the Windows Taskbar to indicate that the translator module 368 is in operation. Other computer systems 110 and architectures may also be used in accordance with the present invention, as the interaction between the operating system 354 and

the device drivers 358 is not critical to the operation of the present invention. (Sim, col. 6, lines 15-25.)

...

FIG. 5a is a flowchart illustrating the processing of assigning commands in accordance with the present invention. First, the user 340 must activate 500 the keyset module 364. The module 364 then displays 504 a list of keymap files 362 to edit and an option to create a new keymap file 362. A table 370 containing the names of the keymap files 362 is located on disk 316, and is accessed to create the list. The system determines 508 if the create new keymap file option is selected. If the create new keymap file option is chosen, then a window allowing the user 340 to search for one of the application modules 366 is created 512. The user 340 selects an application module 366 to be controlled by the game pad 100, and the keyset module 364 receives 516 the input. A new row on the table 370 is created, with the identification of the application module 366 stored as an entry. (Sim, col. 6, lines 34-48.)

Applicants respectfully submit that the passages cited by the examiner fail to teach or suggest “an application program interface that receives calls from the application, the application program interface including a call that creates an association between actions in the application and the controls on the input device, wherein creating the association comprises considering semantics related to the actions in the application.” Instead, Sim is an example of a problem present in the prior art that the claimed arrangement overcomes. A user “must activate 500 the **keyset module 364**” (emphasis added, col. 6, line 36), “a window allowing the user 340 to search for one of the application modules 366 is created 512” (col. 6, lines 42-44), and the “user 340 selects an application module 366 to be controlled by the game pad 100, and the keyset module 364 receives 516 the input” (col. 6, lines 44-46). Further, Sim is clear that the “keyset module” is not the “application module” (col. 6, lines 36-37, 45-46, and Figure 3, items 364 and 366). In the described arrangement, the “application” never calls an API to create “an association between actions in the application and the controls on the input device.” Sim fails to teach or suggest this “call” received “from the application” by the API that creates “the association ... considering semantics related to the actions in the application.”

Additionally, the Office asserts that the recited arrangement is disclosed in the following passages:

The keymap file (362) provides programmable mapping between game pad buttons (102) and application modules and DVD-player commands. In one embodiment, when the present invention detects that an application module (366) is activated, it loads the corresponding keymap file (362) into memory (312), thus allowing the game pad (100) to seamlessly control the application module (366) using the user-defined commands. Multiple keystrokes or commands may be assigned to each button (102), and multiple keymap files (362) may be created for each application module (366). (Sim, col. 2, lines 1-10).

...

Multiple keymap files 362 are coupled to a translator module 368. These keymap files 362 are used by the translator module 368 to translate commands transmitted by the game pad 100 into commands that control specific application modules 366 or devices such as DVD-player 332. For example, if user 340 has a DVD-Player 332 installed in the computer system 110, then commands entered into the game pad 100 are translated by the translator module 368 using the appropriate keymap file 362 into commands recognizable by the DVD-player 332. If user 340 has set button A to represent the "Play" command, for example, and "Play" is initiated by the keyboard button "P," then when user 340 presses button A, the button A signal is translated into the "P" keyboard command, and is sent to the computer system 110 which routes the "P" command in its normal process to the DVD-player 332, which will then begin to play the disk, as described below. Similarly, the keymap files 362 may also be used to define commands for specific games, or other application modules 366. (Sim, col. 5, lines 25-43.)

...

Other buttons 102 on the game pad 100 may be programmed for other functions defined by the user 340. Thus, when user 340 wants to have mouse control of an application module 366, the user 340 can simply assign this mouse emulation keymap file 367 to the application module 366. The user 340 may reprogram the mapping in the mouse emulation keymap file 367 to suit the user's preferences. (Sim, col. 7, lines 35-40.)

Applicants respectfully submit that the passages cited by the examiner fail to teach or suggest "an application program interface that receives calls from the application, the application program interface including a call that creates an association between actions in the application and the controls on the input device, wherein creating the association comprises considering semantics related to the actions in the application."

Applicants do not dispute that Sim discloses that a “keymap file (362) provides programmable mapping between game pad buttons (102) and application modules” (col. 2, lines 1-2). Additionally, Applicants do not dispute that Sim discloses that (i) “keymap files (362) may be created for each application module (366)” (col. 2, line 10), (ii) “keymap files 362 are used by the translator module 368 to translate commands transmitted by the game pad 100 into commands that control specific application modules (col. 5, lines 27-30), and (iii) a “user 340 may reprogram the mapping in the mouse emulation keymap file 367 to suit the user’s preferences” (col. 7, lines 39-40). Finally, Applicants do not dispute that once the “keyset module” creates a keymap (col. 8, lines 3-4; col. 6, line 36), the keymap allows the “game pad (100) to seamlessly control the application module” (col. 2, lines 6-7).

However, Sim fails to teach or suggest that the “application” calls an API to create “an association between actions in the application and the controls on the input device.” Sim fails to teach or suggest a “call” received “from the application” by the API that creates “the association ... considering semantics related to the actions in the application.” Rather, the user “must activate 500 the keyset module 364” (col. 6, line 36), “a window allowing the user 340 to search for one of the application modules 366 is created 512” (col. 6, lines 42-44), and the “user 340 selects an application module 366 to be controlled by the game pad 100, and the keyset module 364 receives 516 the input” (col. 6, lines 44-46). Further, Sim is clear that the “keyset module” is not the “application module” (col. 6, lines 36-37, 45-46, and Figure 3, items 364 and 366). In Sim, the application modules (Sim, Figure 3, item 366) are searched and selected by the user via the keyset module (Sim, Figure 3, item 364; col. 6, line 3, lines 42-44, and lines 44-46). Thus, the “application” never calls an API to create “an association between actions in the application and the controls on the input device.”

Therefore, claim 1 should now be in condition for allowance. Claims 2-21, which depend on claim 1, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over McCauley

Claim 1 is directed to a system for mapping an input device’s controls with an application and recites in part, “an application program interface that receives calls from the application, the application program interface including a call that creates an association between actions in the

application and the controls on the input device, *wherein creating the association comprises considering semantics related to the actions in the application.*”

McCauley fails to teach or suggest at least this feature of claim 1.

McCauley is generally directed to a method and apparatus for interfacing multiple peripheral devices to a host computer. More specifically, McCauley discloses a universal game computer interface ("UGCI") that includes an operating system that enables the UGCI to accept peripheral data packets and signal data, format the received data packets and signal data into human input devices ("HID") report descriptors and HID reports, and transmit the HID reports over a USB cable to the host PC. (McCauley column 6, lines 1-6). The interface controller also transmits descriptors corresponding to each peripheral device to the host PC. (McCauley column 7, line 67 to column 8, line 2).

The Office Action alleges that McCauley teaches the features of claim 1. Specifically, the Office Action states “[w]hen an input device is plugged into the arcade game system, it is identified as a specific type of device (i.e. trackball), and the interface system thereafter generates a trackball report which is an association between actions in the game application and the controls of the input device.”

The report generated in McCauley that the action alleges is an association does not consider any “semantics related to the actions in the application,” as recited in claim 1.

McCauley describes the report as:

a self-descriptive formatted data packet, such as a HID report descriptor, that describes a particular peripheral device. The formatted self-descriptive data packet relates the personality or archetype of the originating peripheral device to a particular archetype model structure contained in the software module, and typically further describes individual design elements, and the nature and relatedness of the elements, of the peripheral device within the context of the referenced archetype model. (Col. 3, lines 35-44.)

Thus, the reports in McCauley contain information related to the device that enables a host computer to communicate with the device. However, the information is related to the device and does not contain any information, such as semantics, *related to the actions in an application*. By contrast, claim 1 recites creating an association between actions in the application and the controls on an input device by “considering semantics related to the actions

in the application,” as recited in claim 1. For example, page 4, lines 4-10 of the application explains:

Each semantic in the driving game genre represents an abstract action that a driving game may be able to perform, such as “steer,” “accelerate,” and “decelerate.” A steering wheel device may correlate the “steer” semantic with turning the steering wheel, and the “accelerate” and “decelerate” semantics with the right and left pedals. A driving game application may correlate the “steer,” “accelerate,” and “brake” semantics with the game actions of turning, speeding up, and slowing down, respectively. The Mapper API maps each device control into the game action associated with the same semantic. The Mapper API uses these correlations to map device controls into software actions; for example, the steering wheel maps to the action of turning the car, and the right and left pedals map to the actions of speeding up and slowing down the car.

In other words, the association created in claim 1 between the controls of the device and actions of the application takes into consideration semantics related to the action of the application, such as “steer,” “accelerate,” and “decelerate” in the driving application example above. McCauley does not teach or suggest any such feature. Therefore, claim 1 should now be in condition for allowance. Claims 2-21, which depend on claim 1, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over Chan

In general, Chan is directed to a system and method for interfacing peripheral devices to a computer universal serial bus. The interfacing system includes a serial EEPROM that is accessed to map sensed key locations for one or more keys into the value that is to be transmitted to the USB. A keyboard-processing capability maps vendor-specific keycodes of the detected keys to their corresponding HID values, which are USB-standardized keycodes, using vendor-supplied tables stored in the EEPROM. (Chan column 5, lines 14-20). Chan is primarily concerned with transmitting standardized USB keycodes from a variety of peripheral devices connected to the USB.

The Office Action alleges that Chan suggests all features of claim 1. Specifically, the Office Action states “when the programs inherently call for the input key combinations, it uses

the association between actions in the application and the input key combinations, as detailed by the information stored in the EEPROM.”

The information that the Office Action alleges to be association information in Chan is described as “configuration, vendor, and mapping sequences which enable the processor to operate rapidly to map one or more actuated keys to a data output representing the operator’s command. The serial EEPROM also operates to exchange information from the peripheral device to the host with a unique handshake protocol which prevents conflicts when accessing the EEPROM.”

The information on the EEPROM in Chan is a table that translates key locations to a corresponding standardized USB keycode to enable the device to communicate with a host computer. However, the information in Chan is specific to the peripheral device and a USB protocol. The “configurable parameter sets, for the keyboard, ps/2 device, and other various I/O devices,” “vendor specific input device codes,” “HID values, which are standardized keycodes,” etc. contained in the EEPROM are not representative of the actions of an application. The information in the EEPROM is simply used to establish communication. None of this information, even given its broadest interpretation, could be construed to suggest to one of skill in the art “semantics related to actions in an application,” such as “steer,” “accelerate,” and “decelerate” in the driving application example previously discussed. Therefore, Chan does not teach or suggest creating an association between actions in an application and the controls on an input device by “considering semantics related to the actions in the application,” as recited in claim 1.

Since the Chan fails to show at least one feature of claim 1, claim 1 should now be in condition for allowance. Claims 2-21, which depend on claim 1, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over Sim

Claim 22 is directed a method for communicating between an input device and an application in a system and recites:

- (a) issuing, from the application, a call to enumerate a suitability of input devices installed in the system, the call including an array of actions that the application uses;

- (b) in response to the application call, examining the input devices installed on the system by comparing controls on the input devices with actions used by the application;
- (c) ranking the input devices based on the comparison; and
- (d) providing the application with at least the highest ranked input device that most closely matches the actions of the application.

The Office Action alleges that Sim discloses the features of claim 22 at various places in col. 5 and col. 6, as follow,

In the preferred embodiment, the drivers 404, 408, 412, 422 communicate with the DirectInput interface 430 developed by Microsoft Corporation. The Direct Input interface 430 specifies a specific protocol for input devices 100, 336 to comply with in order to communicate with an application module 366 or operating system 354. More specifically, the input devices 100, 336 communicate with a DirectInputDevice object created for each input device 100, 336. The function CreateDevice is defined in Direct Input 430 as the function used for creating DirectInputDevice objects. In order to create a DirectInputDevice object, a GUID (Global Unique Identifier as defined by Microsoft.TM. Corporation) must be defined for the input device 100, 336. For game pad 100, a GUID must be defined for the game pad 100 as a joystick. In an embodiment where the game pad 100 creates its own custom mouse emulation module 412, a GUID must be created for the mouse emulation module 412 as well. Currently, DirectInput 430 predefines two GUIDs, GUID_SysKeyboard, and GUID_SysMouse. Thus, when creating the mouse emulation module 412, the GUID_SysMouse must be included. (Col. 5, lines 45-65.)

...

If the computer system 110 already has a mouse 403 as an input device 336, then the EnumDevices is used to enumerate the mouse embodiments of the game pad 100. The parameter DIDEVTYPE_MOUSE is used to provide a mouse GUID for the game pad 100.

After DirectDeviceObjects are created for game pad 100, a data structure for the game pad 100 must be specified. The data structure controls a plurality of aspects of the input device, including axis information, whether relative or coordinate information should be used, as well as other parameters of input devices 336 known to those skilled in the art. For the game pad 100, the data format structure includes the range and motion parameters of the game pad 100. A pointer for the data structure is used to keep track of the multiple button keymapping of the game

pad 100. The structure information is saved as bin format in the keymap files 362. Once the translator module 368 and the keymap files 362 are stored in memory, the game pad 100 is able to control any application module 366 using the DirectInput interface 430. In the preferred embodiment, after installation of the translator module 368, an icon is shown in the Windows Taskbar to indicate that the translator module 368 is in operation. Other computer systems 110 and architectures may also be used in accordance with the present invention, as the interaction between the operating system 354 and the device drivers 358 is not critical to the operation of the present invention. As described herein, in the preferred embodiment, the commands generated by the game pad 100 are translated prior to being transmitted to the operating system 354 or device drivers 358. (Col. 6, lines 1-30.)

Applicants respectfully submit that recited passages, fail to teach or suggest the features of claim 22. There is no application described in Sim that issues a call to “enumerate a suitability of input devices.” There is no call from an application “including an array of actions that the application uses.” There is no “comparing controls on input devices with actions used by the application,” no “ranking input devices,” and no “highest ranked input device.”

Applicants do not dispute that Sim discloses that a “keymap file (362) provides programmable mapping between game pad buttons (102) and application modules” (col. 2, lines 1-2). Additionally, Applicants do not dispute that Sim discloses that (i) “keymap files (362) may be created for each application module (366)” (col. 2, line 10), (ii) “keymap files 362 are used by the translator module 368 to translate commands transmitted by the game pad 100 into commands that control specific application modules (col. 5, lines 27-30), and (iii) a “user 340 may reprogram the mapping in the mouse emulation keymap file 367 to suit the user's preferences” (col. 7, lines 39-40). Finally, Applicants do not dispute that once the “keyset module” creates a keymap (col. 8, lines 3-4; col. 6, line 36), the keymap allows the “game pad (100) to seamlessly control the application module” (col. 2, lines 6-7).

However, Sim fails to teach or suggest an application issuing a call “including an array of actions that the application uses,” in order to “enumerate a suitability of input devices.” Further, there is no “comparing controls on input devices with actions used by the application,” no “ranking input devices,” and no “highest ranked input device.”

Since Sim fails to teach or suggest at least one feature of claim 22, claim 22 should now be in condition for allowance. Claims 23-26, which depend on claim 22, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over McCauley

Claim 22 is directed a method for communicating between an input device and an application in a system and recites:

- (a) issuing, from the application, a call to enumerate a suitability of input devices installed in the system, the call including an array of actions that the application uses;
- (b) in response to the application call, examining the input devices installed on the system by comparing controls on the input devices with actions used by the application;
- (c) ranking the input devices based on the comparison; and
- (d) providing the application with at least the highest ranked input device that most closely matches the actions of the application.

The Office Action alleges that McCauley discloses the features of claim 22 in various places in col. 3 and col. 4. Specifically, the Office Action states,

Wherein ranking is on the basis of HID class or type of the peripheral device sensed, further wherein HID report descriptors corresponding to the device are created and transmitted to the host computer during the enumeration process cycle. A highest ranked input device is signified by the sensed and identified input device causing HID report descriptors to be produced of a particular ranked archetype when the reading of peripheral state information may be indicative of instantaneous input device actuation.

Applicants strongly disagree with these assertions and respectfully submit that there are absolutely no passages in McCauley to support such assertions. There are no passages in McCauley that would suggest any ranking of peripheral devices. Similarly there are no passages that suggest a “highest ranked input device.”

The Office Action cites col. 3, lines 1-15 and col. 4, lines 1-15, as follows, For example, col. 3, lines 1-15 states,

A preferred embodiment of the method of the present invention includes the reading of peripheral state information of certain peripheral devices by an interface control module, where

the peripheral state information may be indicative of an instantaneous mechanical position of an analog potentiometer and/or indicative of an alternately manually depressed and released button. The interface control module transmits formatted data packets, such as HID report descriptors, and formatted signal data, such as HID reports, to the host computer via a communications interface, or bus. The HID report descriptors are substantively formatted according to the HID standard and the archetype and/or structure of the corresponding peripheral device. Each HID report is formatted substantively in accordance with the HID standard and a reading of the state information as read from a specific peripheral device.

This passage contains no teaching or suggestion to rank input devices based on a comparison of controls of a device with actions used by an application, nor any teaching or suggestion of a “highest ranked input device.”

Further, col. 4, lines 1-15, states:

The interface control module operating system is preprogrammed to recognize and expect that a particular HID class or type of peripheral device will be connected to a particular connector of the interface control module. A HID report descriptor corresponding to a HID peripheral archetype, that relates to the expected HID peripheral class or type, is then created and transmitted to the host computer during an enumeration process cycle of the interface control module. Appropriately formatted HID reports are thereafter generated by the interface control module, and transmitted to the host computer, on the basis of the HID class or type of peripheral device and the instantaneous state data read from the peripheral device by the interface control module.

Likewise, this passage also contains no teaching or suggestion to rank input devices based on a comparison of controls of a device with actions used by an application, nor any teaching or suggestion of a “highest ranked input device.” Even if transmitting HID reports to the host computer “on the basis of the HID class or type of peripheral device,” could be construed as suggesting “ranking input devices,” which applicants submit is untrue, the HID class or type of device, is device-specific. The basis for the ranking of input devices in claim 22 is not device-specific. Rather, the basis is a comparison of “controls on the input devices with actions used by the application.”

Since McCauley completely fails to teach or suggest at least one feature of claim 22, claim 22 should now be in condition for allowance. Claims 23-26, which depend on claim 1, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over Chan

The Office Action alleges that Chan teaches “ranking the input devices based on the comparison,” at col. 3, lines 20-30. Specifically, the Office Action states “[e]ach input device, is ranked with a unique address according to the USB standard, and further priority ranked according to a first to be activated, first to be processed system via the serial EEPROM.”

Applicants respectfully disagree with these assertions and submit Chan would not suggest to one of skill in the art to rank input devices based on a comparison of controls of a device with actions used by an application, as recited in claim 22.

Col. 3, lines 20-30 read as follows:

Although the keyboard and mouse have significantly different operative characteristics, the data from both is requested under USB control, verified, buffered, and transferred, with device characteristics being taken into account in building the message frame sequences called for by the USB. To this end, the EEPROM is written for the particular input devices so as to incorporate configuration, vendor, and mapping sequences which enable the processor to operate rapidly to map one or more actuated keys to a data output representing the operator's command. The serial EEPROM also operates to exchange information from the peripheral device to the host with a unique handshake protocol which prevents conflicts when accessing the EEPROM.

In this passage, Chan discloses an interface that receives requests from a host, determines whether the request is made to a keyboard or a mouse, and returns the corresponding data type to the host in an appropriate format. This is accomplished using “configuration, vendor, and mapping sequences” to map actuated keys to a data output consistent with the USB protocol. Nothing in this passage could be understood to suggest a ranking of input devices based on a comparison of controls of a device with actions used by an application, as recited in claim 22. Likewise, no other passage in Chan suggests the ranking feature of claim 22.

Since Chan fails to show at least one feature of claim 22, claim 22 should now be in condition for allowance. Claims 23-26, which depend on claim 22, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over Sim

Claim 27 is directed a method for mapping an input device's controls with an application in a system and recites in part, "in response to a request from an application program to create an action-to-control mapping ... reading a structure that includes action values ... defined by the application."

The Office Action alleges that Sim discloses these features in the passages that follow,

Thus, the computer system 110 only has to process the transmitted commands in its conventional manner, and no enhanced functionality or specific architecture is required.

FIG. 5a is a flowchart illustrating the processing of assigning commands in accordance with the present invention. First, the user 340 must activate 500 the keyset module 364. The module 364 then displays 504 a list of keymap files 362 to edit and an option to create a new keymap file 362. A table 370 containing the names of the keymap files 362 is located on disk 316, and is accessed to create the list. The system determines 508 if the create new keymap file option is selected. If the create new keymap file option is chosen, then a window allowing the user 340 to search for one of the application modules 366 is created 512. The user 340 selects an application module 366 to be controlled by the game pad 100, and the keyset module 364 receives 516 the input. (Col. 6, lines 30-45.)

...

As shown in FIG. 5b, if the user 340 chooses to create a keymap file 362 for an application module 366 that already has a keymap file 362 assigned to it, then the keyset module 364 loads the keymap file 362 into memory 312 and displays 536 the buttons that can be assigned along with their current command assignments, as shown in FIG. 6. The keyset module 364 receives 540 the new commands in the same process as described above, and the keyset module 364 displays 544 the option to save the new command assignments over the existing keymap file 362 or as a new keymap file 362 for that application module 366. The keyset module 364 determines 548 if the user 340 selects save as an existing keymap file 362. If so, the keyset module 364 replaces

568 the existing keymap file 362 with the newly created module 362 containing the new command assignments, and overwrites the link to the old keymap file 362 with a link to the new keymap file 362. (Col. 7, lines 50-67.)

...

If the new file option is chosen, then the keyset module 364 receives 552 a new file name. Then the keyset module 364 displays a window in which the user 340 can select an application module 366 to assign the new keymap file 362. The keyset module 364 receives 560 input selecting the application module 366, and determines 364 whether the application module 366 is already in the table. An application module 366 may have multiple links to different keymap files 362. If the application module 366 is in the table 370, a link to the keymap file 362 is added 367 to the existing application module row. If the application module 366 is not on the table 370, the keymap file 362 is stored 566 in the table 370 in a new column corresponding to the row identifying the specified application module 366. In the multiple keymap file 362 situation, however, user 340 must choose which keymap file 362 to apply upon activating the application module 366 or translator module 368. (Col. 8, lines 1-17.)

These passages contain no teaching or suggestion that discloses “in response to a request from an application program to create an action-to-control mapping ... reading a structure that includes action values ... defined by the application.”

Rather, Sim is an example of a problem present in the prior art that the claimed arrangement overcomes. A user “must activate 500 the **keyset module 364**” (emphasis added, col. 6, line 36), “a window allowing the user 340 to search for one of the application modules 366 is created 512” (col. 6, lines 42-44), and the “user 340 selects an application module 366 to be controlled by the game pad 100, and the keyset module 364 receives 516 the input” (col. 6, lines 44-46). In Sim, the application modules (Sim, Figure 3, item 366) are searched and selected by the user via the keyset module (Sim, Figure 3, item 364; col. 6, line 3, lines 42-44, and lines 44-46). Further, Sim is clear that the “keyset module” is not the “application module” (col. 6, lines 36-37, 45-46, and Figure 3, items 364 and 366). Whereas, in claim 27, the “action-to-control mapping” is “responsive to an application program” that defined the “action values.”

Thus, Sim fails to teach or suggest a method for mapping an input device’s controls with an application in a system and recites in part, “in response to a request from an application

program to create an action-to-control mapping ... reading a structure that includes action values ... defined by the application.”

Since Sim fails to show at least one feature of claim 27, claim 27 should now be in condition for allowance. Claims 28 and 29, which depend on claim 27, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over Chan

Claim 27 is directed a method for mapping an input device’s controls with an application in a system and recites in part, “reading a structure that includes action values and action semantics associated with the action values, the action values being defined by the application.”

The Office Action alleges that Chan teaches this feature of claim 27 at col. 3, lines 55-65, and col. 5, lines 14-21.

Col. 3, lines 55-65 state:

The serial EEPROM is accessed to map the sensed key locations for one or more keys into the value that is to be transmitted to the USB. The system is advantageously organized in the processor and engine configuration so that low cost standard and ASIC circuit units may be employed. It accepts the intermittent and relatively low and medium speed inputs from the operator-controlled devices and meets all the USB requirements for identification, error detection and correction, hand shaking, and message transfer. Thus, it is fully consistent with the objectives of the USB in terms of low cost, expandability and achieving true plug and play capability.

This passage contains no teaching or suggestion to read “a structure that includes action values and action semantics associated with the action values, the action values being defined by the application.” The passage is silent as to actions values being defined by the application, as well as any action semantics associated with the action values.

Col. 5, lines 14-21 state:

Subsequent to successfully debouncing a combination of detected keys, the keyboard-processing capability 4 maps the vendor-specific keycodes of the detected keys to their corresponding Human Input Device (HID) values, which are USB-standardized keycodes, using vendor-supplied tables stored in the serial EEPROM 24. In turn, the keyboard-processing capability 4 builds a Keyboard Input Report consisting of the HID values and

formatted according to USB descriptor information obtained from the serial EEPROM 24.

This passage in Chan describes mapping vendor-specific keycodes to their corresponding HID values, which are USB standardized keycodes, using vendor-supplied tables stored in the EEPROM. A report is then created including the proper HID values and formatted according to the vendor-provided table in the EEPROM. However, claim 27 recites, “a structure that includes action values and action semantics associated with the action values, *the action values being defined by the application.*” Neither of the “vendor-specific keycodes (supplied by the vendor of the peripheral device),” or “HID values (USB standardized keycodes),” are “defined by the application,” as recited in claim 27. Thus, Chan could not possibly teach or suggest “reading a structure that includes action values and action semantics associated with the action values, the action values being defined by the application” to one of ordinary skill in the art.

Since Chan fails to show at least one feature of claim 27, claim 27 should now be in condition for allowance. Claims 28 and 29, which depend on claim 27, should be allowable for at least the same reasons, as well as the respective features recited therein.

Patentability over Sim

Claim 30 is directed a computer-readable medium including computer-executable instructions to perform a method for using a computer input device with a software application, and recites in part, “an application program interface, responsive to a call from an application, that returns an enumeration of input devices that substantially match the actions of the application.”

The Office Action alleges that Sim discloses these features in the passages that follow,

In the preferred embodiment, the drivers 404, 408, 412, 422 communicate with the DirectInput interface 430 developed by Microsoft Corporation. The Direct Input interface 430 specifies a specific protocol for input devices 100, 336 to comply with in order to communicate with an application module 366 or operating system 354. More specifically, the input devices 100, 336 communicate with a DirectInputDevice object created for each input device 100, 336. The function CreateDevice is defined in Direct Input 430 as the function used for creating DirectInputDevice objects. In order to create a DirectInputDevice object, a GUID (Global Unique Identifier as defined by Microsoft.TM. Corporation) must be defined for the input device 100, 336. For game pad

100, a GUID must be defined for the game pad 100 as a joystick. In an embodiment where the game pad 100 creates its own custom mouse emulation module 412, a GUID must be created for the mouse emulation module 412 as well. Currently, DirectInput 430 predefines two GUIDs, GUID_SysKeyboard, and GUID_SysMouse. Thus, when creating the mouse emulation module 412, the GUID_SysMouse must be included. (Col. 5, lines 45-65)

...

If the computer system 110 already has a mouse 403 as an input device 336, then the EnumDevices is used to enumerate the mouse embodiments of the game pad 100. The parameter DIDEVTYPE_MOUSE is used to provide a mouse GUID for the game pad 100.

After DirectDeviceObjects are created for game pad 100, a data structure for the game pad 100 must be specified. The data structure controls a plurality of aspects of the input device, including axis information, whether relative or coordinate information should be used, as well as other parameters of input devices 336 known to those skilled in the art. For the game pad 100, the data format structure includes the range and motion parameters of the game pad 100. A pointer for the data structure is used to keep track of the multiple button keymapping of the game pad 100. The structure information is saved as bin format in the keymap files 362. Once the translator module 368 and the keymap files 362 are stored in memory, the game pad 100 is able to control any application module 366 using the DirectInput interface 430. In the preferred embodiment, after installation of the translator module 368, an icon is shown in the Windows Taskbar to indicate that the translator module 368 is in operation. Other computer systems 110 and architectures may also be used in accordance with the present invention, as the interaction between the operating system 354 and the device drivers 358 is not critical to the operation of the present invention. As described herein, in the preferred embodiment, the commands generated by the game pad 100 are translated prior to being transmitted to the operating system 354 or device drivers 358. (Col. 6, lines 1-30.)

These passages fail to teach or suggest an “application program interface, responsive to a call from an application, that returns an enumeration of input devices that substantially match the actions of the application.”

Instead, Sim is an example of a problem present in the prior art that the claimed arrangement overcomes. A user “must activate 500 the **keyset module 364**” (emphasis added, col. 6, line 36), “a window allowing the user 340 to search for one of the application modules 366 is created 512” (col. 6, lines 42-44), and the “user 340 selects an application module 366 to be controlled by the game pad 100, and the keyset module 364 receives 516 the input” (col. 6,

lines 44-46). In Sim, the application modules (Sim, Figure 3, item 366) are searched and selected by the user via the keyset module (Sim, Figure 3, item 364; col. 6, line 3, lines 42-44, and lines 44-46). Further, Sim is clear that the “keyset module” is not the “application module” (col. 6, lines 36-37, 45-46, and Figure 3, items 364 and 366). Further, the recited passages make no mention of “an enumeration of input devices that substantially match the actions of the application.”

Thus, Sim fails to teach or suggest an “application program interface, responsive to a call from an application, that returns an enumeration of input devices that substantially match the actions of the application.”

Since Sim fails to show at least one feature of claim 30, claim 30 should now be in condition for allowance. Such action is respectfully requested.

Patentability over Chan

Claim 30 is directed a computer-readable medium including computer-executable instructions to perform a method for using a computer input device with a software application, and recites in part, “an application program interface, responsive to a call from an application, that returns an enumeration of input devices that substantially match the actions of the application.”

The Office Action alleges that “an API, responsive to a call from an application, that returns an enumeration of input devices that substantially match the actions of the application,” is taught by Chan at column 5, lines 35-65. This passage in Chan discloses a PS/2 device processing capability that processes data from the host and makes it available to the PS/2 device and a host-processing capability that waits for device data requests from the host. (Chan column 5, lines 15-50). The host-processing capability receives requests from a host, determines whether the request is made to a keyboard or a mouse, and returns available device reports. (Chan column 5, lines 35-50). Chan, col. 5, lines 50-65 further explains that the device reporting capabilities in the USB interface can also report the availability of numerous other devices.

However, Chan is concerned with the type of device that is being connected (mouse, keyboard, serial devices, parallel devices, etc) such that data from each of the devices can be “requested under USB control, verified, buffered, and transferred with device characteristics being taken into account in building the message frame sequences called for by the USB.” See

Chan, col. 3, lines 19-23. This process is silent as to "actions of the applications." The actions of the applications are irrelevant to a process for translating between the individual characteristics of the input device and the USB protocol. Therefore, this process does not disclose "an application program interface, responsive to a call from an application, that returns an enumeration of input devices that substantially match the actions of the application."

Since Chan fails to show at least one feature of claim 30, claim 30 should now be in condition for allowance. Such action is respectfully requested.

Patentability of Claim 27 over McCauley in view of Chan

The Office Action rejects claim 27 under 35 U.S.C. § 103(a) as unpatentable over McCauley in view of Chan. Applicants respectfully submit the claims in their present form are allowable over the cited art. To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. (MPEP § 2142.) Motivations to combine or modify references must come from the references themselves or be within the body of knowledge in the art. (See MPEP § 2143.01.) The rejection is respectfully traversed.

As discussed previously with respect to claim 27, Chan fails to teach or suggest "reading a structure that includes action values and action semantics associated with the action values, the action values being defined by the application."

McCauley fails to remedy the deficiencies of Chan. As explained previously, McCauley discloses a universal game computer interface ("UGCI") that includes an operating system that enables the UGCI to accept peripheral data packets and signal data, format the received data packets and signal data into human input devices ("HID") report descriptors and HID reports, and transmit the HID reports over a USB cable to the host PC. (McCauley column 6, lines 1-6). The interface controller also transmits descriptors corresponding to each peripheral device to the host PC. (McCauley column 7, line 67 to column 8, line 2).

At col. 3, lines 35-44, McCauley describes construction of a device-specific data structure from a self-descriptive formatted data packet (HID report descriptor), "that describes

the particular peripheral device. The formatted self-descriptive data packet related the personality or archetype of the originating peripheral device to a particular archetype model structure contained in the software module, and typically further describes individual design elements, and the nature and relatedness of the elements, of the peripheral device with in the context of the referenced archetype model.” Similar to Chan, McCauley only describes device-specific information. Neither the formatted data packet, nor the data structure constructed based on the packet, contain any information pertinent to “action values” defined by the application or “action semantics associated with the action values.”

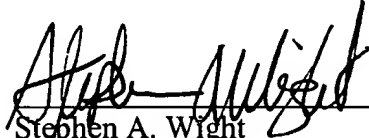
Since the applied art, either alone or in combination, fails to show at least one feature of claim 27, claim 27 should now be in condition for allowance. Claims 28 and 29, which depend on claim 27, should be allowable for at least the same reasons, as well as the respective features recited therein. Such action is respectfully requested.

Conclusion

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By 
Stephen A. Wight
Registration No. 37,759

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446
(154789.2)